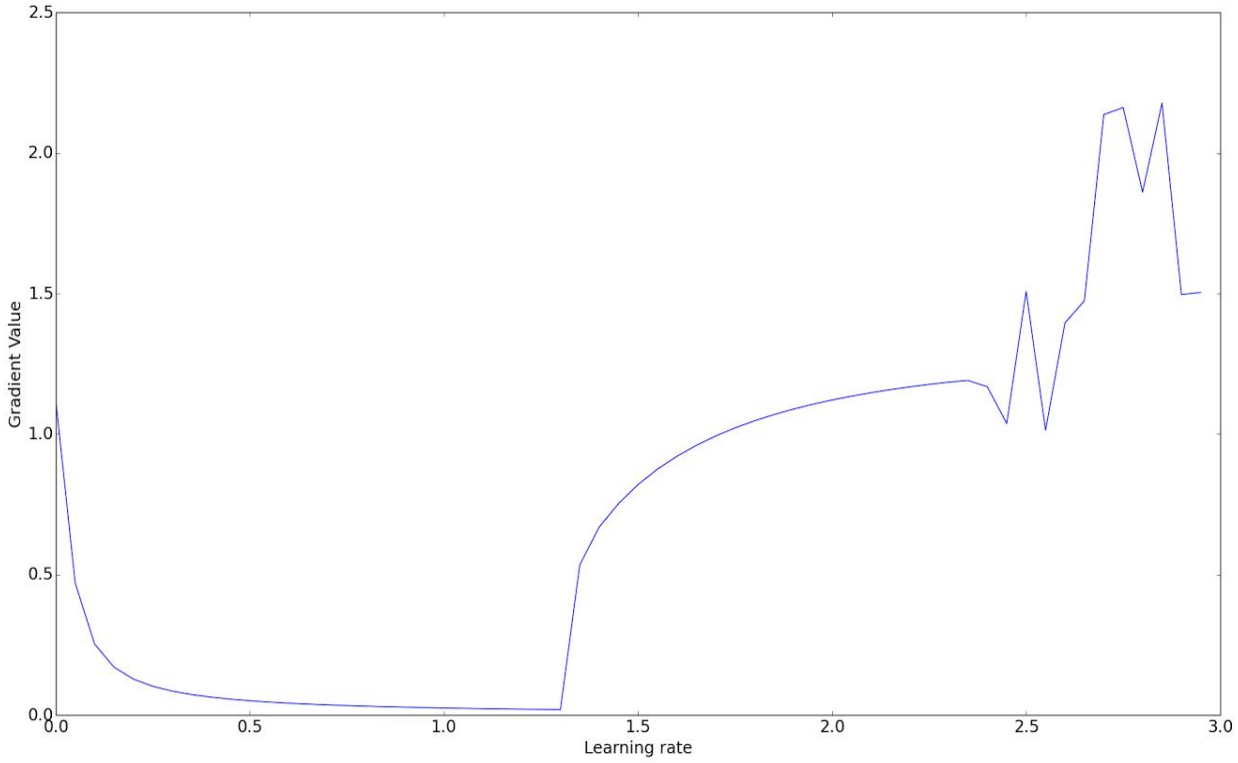
Assignment-2: Maximum Entropy Policy

Submitted By: Ajinkya Jain

Q*1. Experimentally choose a static step size / learning rate for gradient descent (i.e. a scalar value to multiply the gradient by, in order to decide on the magnitude of the weight update at each epoch). Describe how you chose an appropriate learning rate.*

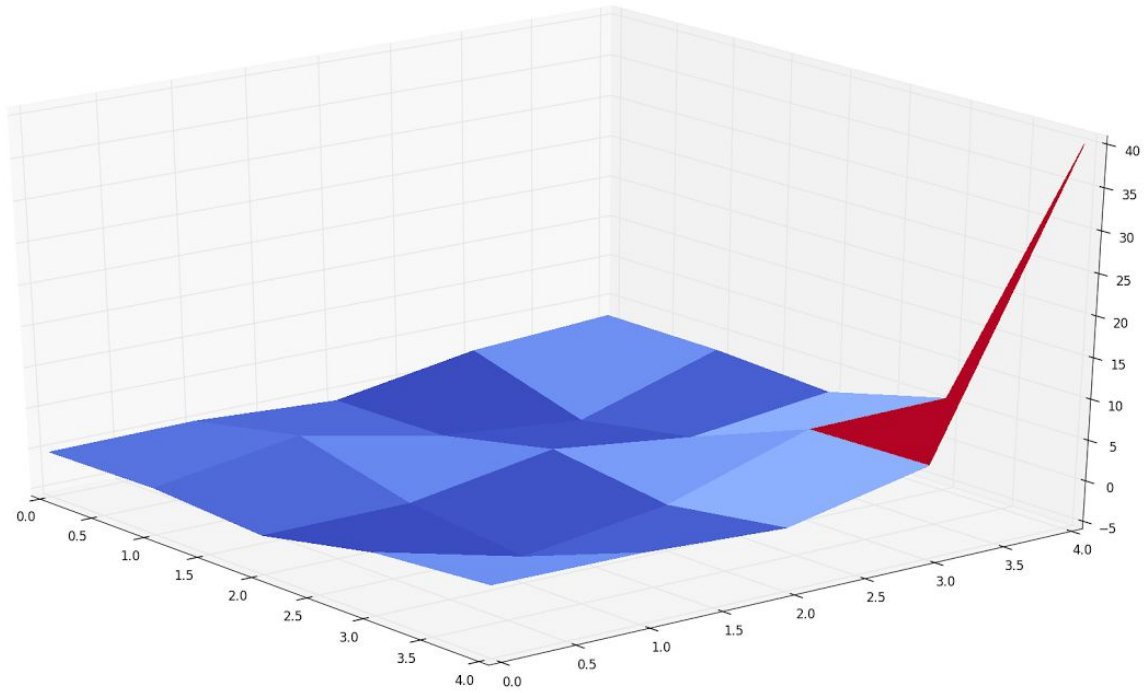
⇒

Optimal value of the learning rate is the one that leads to the convergence of the results fastest. This can also be interpreted as, the optimal learning rate is the one that reduces the gradient update to zero fastest. Considering a fixed number of gradient descent steps (n_epoch = 100), the learning rate that leads to maximum reduction in the value is the optimal one. Based on this interpretation, the learning rate is varied from 0.0 to 3.0 with a step size of 0.05. The plot is as shown below. From the plot it can be seen as the optimal learning rate should be around 1.3 as leads to the most reduction in the value of the gradient.



Q2. *Run MaxEntIRL for 100 epochs with a horizon of 10 and describe the shape of the reward function. What does it appear to be trying to do, qualitatively? How does it behave at state 8 and 16? Does it appear to underfit or overfit the demos? If so, how might this issue be addressed, while still using the same underlying algorithm? Include a graph of the reward function in your writeup.*

⇒
The reward function have a ridge-like shape connecting the start state [0] and the terminal state [24] diagonally. The states 8 and 16 lie at the bottom of the valley representing the lowest rewards. The reward function assigns a very high reward to state 24.



Qualitatively, the reward function is shaped to represent the most optimal path connecting the start to the end goal, which would be following a diagonal path as close as possible. It is trying to avoid the states 8 and 16 while traversing along the shortest path between the start and the goal position.

If the model is underfitting the data, then the model performs poorly on the training data. If the model is overfitting the data, it performs well on the training data but performs badly on evaluation data. In this case, as we can see from the plot fig2, the model is performing sufficiently good on the training data. Hence, the model is not underfitting the demos.
The model is giving extremely low rewards to state 16 as none of the demonstrations visit it, which means that the model is overfitting the demonstrations.
The problem of overfitting can be resolved either by providing more demonstrations or using better features that take in account whether there is a hazard at a state or not etc.

Q3. *Call calcMaxEntPolicy() with a specific reward function: +10 at state 24 and -1 everywhere else with a horizon of 10. Do NOT perform gradient descent or call any other parts of the MaxEntIRL algorithm—just call the calcMaxEntPolicy() function with the given reward functions. Compare and contrast the returned policy with the optimal policy under the same reward function. In what ways is it different and why? In particular, examine the most likely action at State 19 under the MaxEnt policy and explain the rationale for it.*

⇒ The new policy for the reward function will look like as shown below. (Ties are broken at random)

↓	↓	↓	↓	↓
→	↓	↓	↓	↓
→	→	↓	↓	↓
→	→	→	→	←
→	→	→	↑	↓

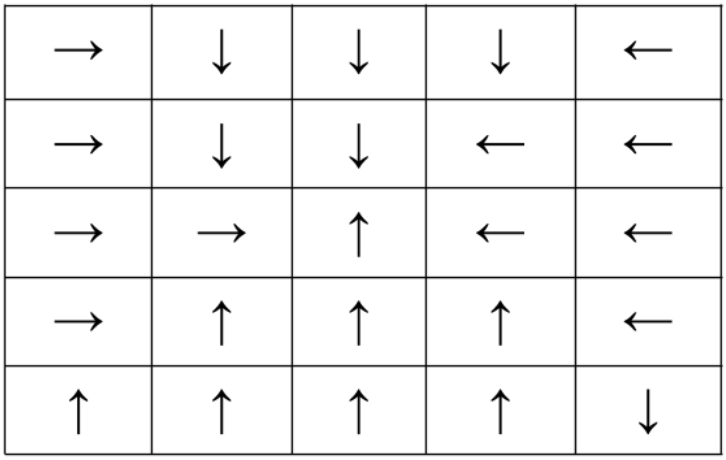
The optimal policy for the current reward function will look like as shown below

↓	↓	↓	↓	↓
→	↓	↓	↓	↓
→	→	↓	↓	↓
→	→	→	↓	↓
→	→	→	→	↓

As can be seen from the two figures, the two policies are mostly similar except for differences in a few states.
The most likely action at state 19 under MaxEnt policy is to go left, whereas the optimal action to take will be to go down. This deviation from the optimal behaviour is because the MaxEnt policy assigns equal probability to all the trajectories having equal rewards. From state 19, there is only one trajectory that leads to the terminal state, whereas, there are many more slightly

suboptimal trajectories that lead to the terminal state. Hence, under MaxEnt policy, the optimal policy, even though it generates the maximum reward, will become less probable and the slightly suboptimal policy will become the dominant trajectory to take.

Q4. *Increase the horizon to 100 for the same reward function as above and call calcMaxEntPolicy() again. What happens to the MaxEnt policy and why? If it is difficult to interpret all the probabilities of actions at each state, try just looking at the most likely action at each state*.



⇒
The MaxEnt policy is shown as above (with most likely action shown). As we increase the horizon, we are increasing the number of trajectories under consideration. Following by the same logic as we have used in the previous question, we can see that the number of suboptimal trajectories have increased and the optimal policy has become even less probable. Hence, we can see the effect we observed in the state 19 in the previous question have propagated to other states of the system and hence the MAxEnt policy is generating even more suboptimal policies as the most probable ones.

Q5. *Compare equations 5 and 7. Intuitively, why does the MaxEnt policy of equation 5 equally weight paths in the example in Figure 2, while the action based distribution model of equation 7 weights them differently? Why is the weighting of equation 7 problematic?*

The action based distribution model of equation 7 will assign a probability distribution over different trajectories originating from that action based on the local information of the action-values. At each node, it will assign equal probabilities to all the paths having same Q-value. Hence, even in the case of paths having equal total returns, it will assign higher probability to the paths that branch earlier in comparison to the paths that branch later.
On the contrary, the MaxEnt policy generates the distribution over complete paths (and not at each node) based on the global information of the total returns obtained by the path. Hence, it will assign equal probabilities to even those path that branch at different times but have the same total path rewards.

The weighting of equation 7 is problematic, as it biases the trajectories taken from an action towards the paths that have early branching. This biasing can result in choosing a suboptimal policy that branched earlier instead of an optimal policy that leads to higher rewards.

Q6. *Why is Zs set to one for terminal states? What is interpretation of this in terms of both 1) probability of trajectories that end at each terminal state and 2) the reward gained for being in a terminal state? (hint: think of the equation for Zaij as a state-action value function)*

⇒

(a) In terms of the probability of the trajectories ending at the terminal state:

Z_{s_k} represents the unnormalized probability of all the trajectories on which s_k lies. In the case of terminal states, all the length 1 trajectories starting at the terminal state must end in the terminal state. Out of all the other trajectories having path lengths from 2 to the length of horizon, some/all (some in the case of multiple terminal trajectories and all in the case of a single terminal state) of them will end at this terminal state . This combined unnormalized probability of the trajectories ending at this terminal state will make the $Z_{s_{Terminal}}$ a non-zero positive number. As this is the only constraint on the value of $Z_{s_{Terminal}}$, to better represent it as a probability, it can be chosen as 1.

(b) In terms of the reward gained for being in a terminal state:

The action-value function is defined as $$Q_(s,a) = \sum_k P(s_k|s_i, a_{i,j})(R(s_i) + V(s_k|s_i, a_{i,j}))$$

In the case of terminal state, as all the further actions are identical and lead to the same transition back to the terminal state itself, they can be considered as a single action. Therefore, we can say that $Z_{s_k} = Z_{a_{ij}}$. If we assume $Z_{s_{Terminal}} = e^{V(s_{Terminal})}$, then $Z_{a_{Terminal,j}} = e^{V(s_{Terminal})}$ too. Therefore, the partition function for the state-action pair can be seen as a state-action value function as following

$$Z_{a_{i,j}} = \sum_k P(s_k|s_i, a_{i,j}) e^{reward(s_i|\theta)} Z_{s_k}$$

$$Z_{a_{ij}} = \sum_k P(s_k|s_i, a_{i,j}) * e^{reward(s_i|\theta)} * e^{V(s_k)}$$

$$Z_{a_{Terminal,j}} = \sum_k P(s_k|s_i, a_{i,j}) * e^{reward(s_i|\theta) + V(s_k)}$$

As the terminal states can transition only to itself, hence $\sum_k P(s_k|s_i, a_{i,j}) = 1$. Therefore,

$$Z_{a_{Terminal,j}} = e^{V(s_{Terminal})} = e^{reward(s_{Terminal}|\theta) + V(s_{Terminal})}$$

If the reward in the terminal state = 0, then the assumption of $Z_{s_{Terminal}} = e^{V(s_{Terminal})}$ satisfy the equation. In general, the value of the terminal state $V_{s_{Terminal}}$ is set to be 0. Therefore, the value of $Z_{s_{Terminal}} = e^{V(s_{Terminal})} = e^0 = 1$